

Optimal Queue Length in Torus Networks with Traffic Signals

Oscar Morales-Ponce and Burkhard Englert and Mehrdad Aliasgari

Department of Engineering and Computer Science,
California State University Long Beach

[oscar.morales-ponce, burkhard.englert, mehrdad.aliasgari]@csulb.edu

Abstract

Inspired by the need for efficient traffic control in urban areas, we study the problem of minimizing the longest queue length in traffic networks. In our model cars only move forward and do not overtake each other and time is divided in rounds of a given length. Cars always cross intersections when they have the green signal and they are at the front of the queue, otherwise they wait. Each car can cross at most one intersection in each round. The only means to control the movement of the cars is by scheduling the green times at the junctions, that allow cars to cross. The objective is to determine the green time schedules to reach a pattern that attains the social optimum, i.e., the minimum possible longest queue length in the network.

We study this problem in an oriented torus of dimension $n \times n$ where each horizontal and vertical ring has kn cars arbitrarily deployed. We introduce the concept of conflict graphs that represent the propagated negative impact across the network when -through green time assignments - the queue length in one segment is reduced. Using these graphs, we provide a lower bound on the longest queue length that any algorithm can attain and show that maximizing the number of cars that cross the intersections in every round can prevent reaching the minimum possible longest queue length. Next, we propose a quadratic running time algorithm that minimizes the longest queue length in each round assuming that the number of cars in each queue is at least the size of the round. Our technique is based on network flooding on the conflict graphs. Finally, we propose an algorithm that guarantees that the cars eventually form a social optimum queue length at the expense of having some green times where no cars cross an intersection.

1 Introduction

Inspired by the traffic control problem in urban areas and with the goal to mitigate traffic jams, we study the problem of minimizing the longest queue length in road networks with a set of cars whose movements are only controlled by traffic signals at intersections. In our context, the problem aims - through the use of traffic signals - to coordinate the set of cars to get into and maintain a formation that attains a social optima, namely that minimizes the longest queue length in the network. Minimizing the longest queue consists of finding a solution that has the best worst case queue length over all possible initial car deployments.

We consider a deterministic approach in symmetric networks in a finite region in which each intersection consists of *two* incoming and *two* out-going road segments or *links* with wrap around. In other words, the links form an oriented torus with n horizontal rings and n vertical rings ($n \times n$ intersections). Thus, the number of cars in the network overall and in each ring remains the same at all times. Hence, the system is

closed. We simply refer to these networks as an oriented torus of dimension $n \times n$ and to the rings as roads. The motivation behind the closed system model is that it allows us to first study the underlying problem and to focus on understanding the dependence of queue lengths on green time assignments at adjacent traffic signals, as well as on the hidden influence of assignments at roads that are far away from each other.

We assume that cars move through the network in rounds. Each intersection (also known as *junction*) has a traffic signal that assigns non-overlapping green times to the incoming roads in each round (called a *phase* in the traffic control literature) of ψ time units according to a given schedule.

Our goal is to analyze the impact of green time assignments at individual intersections on the system as a whole. To enable this analysis we restrict the movement of cars and fix the time it takes for them to move through intersections and links. In our model, agents or cars are stubborn and perpetually follow the same road, that is they never change direction or move onto a different road. They only move forward and do not overtake each other. Agents can cross at most one intersection in each round. We assume that the time to traverse a link between traffic signals in free flow is negligible. An agent always crosses when it is at the front of the queue and it has the right to cross. We assume that agents cross an intersection in exactly one unit of time. Hence, the total number of agents that cross each junction in one round is at most ψ . In urban scenarios, this value is given by the expected maximum number of cars that can cross during a round (a.k.a. as *saturation flow* in the traffic control literature).

Let A denote the set of green time assignments and let $W(r)$ be the agent deployment at round r . We denote by $w_e(W(r), A)$ the number of agents on link e resulting from applying the green time set assignment A to the current deployment $W(r)$. The first problem that we study is to determine the green time assignment A that minimizes the longest queue length in every round in the network. Formally.

Problem 1. *Given a round r and an oriented torus \mathcal{T} and a set of arbitrarily deployed agents W on \mathcal{T} , determine the green time assignment A that minimizes the longest queue length in the next round. Let ϕ denote the optimal queue length. Then,*

$$\phi = \min_{\forall A} \max_{\forall e \in E} \{w_e(W(r), A)\}.$$

In Section 4 we propose a centralized algorithm that determines the green time assignment that minimizes the longest queue length in each round where the traffic signals allow the maximum number of agents at each intersection to cross. In our approach, we consider a conflict graph of the given traffic network that represents the possible conflicts arising when the queue length in one link is reduced, but as a result the conflicting queue increases. We reduce the queue length of a link by decreasing the green time of its neighbors, which in turn means that the queue length of the neighbors can increase beyond a given value. Hence such a change affects its conflicting neighbors and possibly the neighbors of the neighbors and it is critical that we model such chains of consequences of decisions that are made at one intersection. We refer to this chain effect as *flooding the conflict graph*. We show that any algorithm that allows ψ agents to cross in every intersection, regardless of the direction, can create a conflict cycle that prevents the algorithm from reaching a social optima, namely a minimum longest queue length. It follows directly from this that this problem does not always have an optimal solution in saturated networks, where the number of agents in each link is greater than the round length.

The second problem that we study is the problem of finding the minimum longest queue length possible. Formally:

Problem 2. *Given a round r and an oriented torus \mathcal{T} and an initial set of arbitrarily deployed agents W on \mathcal{T} , determine a strategy \mathcal{A} that provides a set of green time assignment such that*

$$W(r+i+1) = w_e(W(r+i), A(r+i)) \text{ for } i \in [0, k)$$

and

$$\phi^*(r+k) = \min_{\forall A \in \mathcal{A}} \max_{\forall e \in E} \{w_e(W(r+k-1), A(r+k-1))\}$$

is the minimum queue length possible over all strategies for some $k > 0$.

We study Problem 2 in Section 5. We propose a centralized algorithm that determines a finite strategy \mathcal{A} that guarantees that agents form a pattern that represents the social optimum. We show that the optimal solution can be obtained if some intersections in the network allow in some rounds less than ψ agents to cross, regardless of the direction.

The solution of Problem 2 implies that the algorithm converges to the optimal value from any initial configuration in a finite number of steps. In a practical application such as urban traffic networks, the algorithm can be constantly run to adapt to the current conditions.

1.1 Contributions

In this paper, we study the problem of minimizing the longest queue length in torus networks. To solve the problem, we propose to use the concept of conflict graphs in torus networks which represent the interactions and interdependencies across the network that arise when the queues in a set of links are reduced through an increase in green times while the queues in a set of conflicting links are increased. We also provide lower bounds on the queue lengths that any algorithm can attain as well as upper-bounds to minimize the longest queue length. The techniques that we employ are based on network flooding. We implement the algorithm to corroborate its effectivity. To the best of our knowledge, this is the first paper that considers the use of conflict networks to address the problem of minimizing the queue lengths of the torus network as well as providing lower bounds for this problem.

1.2 Model

As usual we represent our network as an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. A graph is 2-edge connected if every edge is in a cycle. An orientation of G is the directed graph obtained by assigning to each link a direction.

We consider a set Ω of agents arbitrarily deployed on the oriented torus $G = (V, E)$ of dimension $n \times n$. An oriented torus is the digraph $G = (V, E)$ where $V = \{v_{i,j} | \forall i \in [0, n-1] \wedge \forall j \in [0, n-1]\}$ is the set of nodes and $E = \{\{v_{i,j}, v_{k,l}\} | (k = i+1 \bmod n \wedge l = j) \vee (k = i \wedge l = j+1 \bmod n)\}$ is the set of links. Thus, the number of vertices is n^2 and the number of links is $2n^2$. Throughout the paper, we refer to the ring formed with links $\{v_{i,j}, v_{i,(j+1) \bmod n}\}$ for all $j \in [0, n-1]$ as the horizontal road i . Similarly, we refer to the ring formed with links $\{v_{i,j}, v_{(i+1) \bmod n, j}\}$ for all $i \in [0, n-1]$ as the vertical road j . To simplify our presentation, we consider a torus where each road has a fixed direction and the road directions alternate between neighboring roads. However, our solutions and analysis can easily be extended to a torus where roads have consistent arbitrary directions. In this paper, we assume that each road has kn agents for some $k > 0$. Thus, $|\Omega| = (kn)^2$. Let $E^{in}(v)$ denote the set of incoming links to v and $E^{out}(v)$ denote the out-going links from v . Given a link $e = (u, v)$, let $succ(e)$, $pred(e)$ denote the successor and predecessor link of e in the same road in the direction of the flow. Two links e, e' incident to a vertex are in conflict if reducing the queue length of e increases the queue length of e' . Let $conf(e)$ denote the forward conflicting link of e , i.e., $\{e, conf(e)\} \in E^{in}(v)$. Let $bconf(e) = succ(conf(pred(e)))$ denote the backward conflicting link of e , i.e., $\{e, bconf(e)\} \in E^{out}(v)$.

We divide the time into rounds (sometimes called *phases* in the traffic signal control literature) of ψ time units. Each vertex v assigns the right of way to its incoming links according to a given green time

assignment. A green time assignment is a partition of the round ψ into the number of incoming links, i.e., each incoming link e has a non overlapping time $g_e > 0$ such that $\sum_{e \in E^{in}(v)} g_e = \psi$. Throughout the paper we assume that ψ is even.

In our model, agents only move forward and do not overtake each other. Thus, agents maintain the order in which they were first deployed and the number of agents remains the same at all time. An agent at the front of the queue in a link e always moves in one time unit to $\text{succ}(e)$ when e has the green time. Otherwise, it waits in e . We assume that the time to traverse a link is negligible. Moreover, we assume that agents cross an intersection in one unit of time. However, agents can cross at most one intersection in every round.

Let $w_e(r)$ be the number of agents in e at round r . We define the agent deployment at time r as $W(r) = \{w_e(r) : \forall e \in E\}$. Throughout the paper, we usually omit the time reference if it is clear from the context.

For the dynamics of the system, we consider the store and forward model.

Definition 1. (Store and forward model) *Given a deployment $W(r) = \{w_e(r) : \forall e \in E\}$ and a set $G(r) = \{g_e(r) : \forall e \in E\}$ of green time assignments, the deployment of $e \in E$ at round $r + 1$ is given by:*

$$w_e(r + 1) := w_e(r) - \min(g_e(r), w_e(r)) + \min(g_{\text{pred}(e)}(r), w_{\text{pred}(e)}(r)).$$

We say that the traffic network is saturated if $w_e(r) \geq \psi$ for all $e \in E$. Thus, we can simplify the store and forward model in saturated traffic networks and rewrite it as $w_e(r + 1) := w_e(r) - g_e(r) + g_{\text{pred}(e)}(r)$.

1.3 Related Work

Problems 1 and 2 are strongly related to traffic control algorithms. The main approach in traffic control is to optimize a global measure such as queue length. Prior work, however, does not consider the mini-max problem. Although, there exists literature on min-max model predictive control [7, 8, 15] also called robust model predictive control, they do not deal with traffic control. Moreover, the model in [7, 8, 15] considers continuous flow that does not fit our model.

There are different studies for optimizing traffic networks: Recently, several papers were published using a control theory approach. Specifically, researchers have focused on model predictive control (MPC). In [21], the authors investigate a strategy based on MPC that is specialized on urban traffic management so that traffic congestion is relieved and the travel time is reduced. In [2], the authors propose a methodology that consists of an open-loop constrained quadratic optimal control problem, whose numerical solution is achieved via quadratic programming. In [4], the problem is studied under three methodologies of the macroscopic model that are based on the store-and-forward model. They first formulate the problem as a linear-quadratic control problem. Then they formulate it as an open-loop constrained quadratic optimal control problem that is solved using quadratic-programming. Finally, they formulate it as an open-loop constrained nonlinear optimal control problem that is algorithmically solved. In [18] the authors reformulate the optimization problem into a mixed-integer linear programming (MILP) optimization problem to increase the real-time feasibility of the MPC control strategy. In [3], the authors investigate the efficiency of a recently developed signal control methodology. The corresponding optimization algorithm is embedded in a rolling-horizon MPC scheme. Unlike our model, all these studies consider models that are continuous.

The previous results are based on centralized algorithms. Distributed algorithms based on MPC have also been proposed. In [11], the authors use MPC to solve the problem and propose a distributed algorithm for solving it. Essentially, they decompose the problem into small subproblems whose solutions interact to solve the larger problem. In [9] the author extends a baseline model by introducing constraints on the output of the subsystems and by letting subsystem dynamics depend on the state besides the control signals of the

subsystems in the neighborhood. Thus, they can constrain the queue lengths and delay dynamic effects. Although, the queue length can be constrained, the solution does not guarantee to converge to a state that achieves the minimum longest queue possible. In [23], the authors propose a distributed MPC approach (DCA-MPC) to coordinate and optimize the signal splits that reduces the computational complexity and improves the applicability of traffic signal control approaches based on MPC in practice. The authors in [20], propose a traffic-responsive optimal signal split algorithm that takes into account the uncertainty to minimize the weighted link queue lengths. They suggest a minimax optimization to obtain the green time assignment, which minimizes the objective function when worst case uncertainty appears. However, the previous result does also not consider minimum longest queue length.

In [17] the authors study a slightly different problem in wireless networks where a greedy approach based on the longest queue is used for scheduling transmissions. This approach was used in [5], along with an artificial intelligence system, called Q-learning, for addressing the traffic congestion. In [1], the authors also use the Q-learning approach to address the traffic congestion problem.

Another approach for solving the traffic congestion problem is based on heuristics, where agents are used to self-organize the traffic signals. In [12] the authors propose a simple heuristic based on multi-agents. They show based on simulations that the traffic signals self-organize and adapt to changing traffic conditions without using direct communication. In [16], the problem is studied based on fluid-dynamics and car-following simulations of traffic flows in traffic networks. They propose a self-organizing solution with multi-agent interactions between vehicles and traffic signals that is based on priorities. The interactions lead to “green wave” patterns that decentralize traffic signal control. They claim that the solution may result in an almost periodic service that suggest the existence of a spontaneous synchronization of traffic signals. However, since the approach is based on a heuristic, the optimality cannot be guaranteed.

Other heuristics based on cellular automata models for city traffic have also been proposed. A cellular automaton (CA) is a discrete model based on simple rules. Essentially, it consists of a grid where each cell has a value. Simple rules can change the value of the cells over time. In [6], three adaptive strategies are proposed that can react to traffic conditions. They show that the solution can attain optimal values in some circumstances. There is, however, no a priori guarantee for optimality. Another heuristic is based on genetic algorithms that can solve optimization problems. For example [13] uses genetic algorithms to address the problem.

The traffic control problem has also been studied using a queuing theory approach. In [19], the authors analyze the vehicle conservation equation along the street based on the dynamics of shock waves of queues at isolated signalized intersections. They provide a criteria for the development of saturated or unsaturated situations, expressions for the maximum queue length and limits on the control variables. In [14], the authors use an approach based on game theory. They present a distributed game which reacts to the traffic conditions.

Although many papers have been published on the problem of mitigating the effect of traffic signals on urban areas under distinct approaches, this is the first work that formally characterizes the lower bounds on the queue lengths that can be obtained in each round.

1.4 Paper Organization

The remaining paper is organized as follows. We introduce the concept of conflict graph that we use throughout the paper in Section 2. Section 3 is devoted to present the lower bounds for the two problems. In Section 4, we present an algorithm that minimizes the longest queue in every round. In Section 5, we present a strategy that guarantees that the agents will form a social optima. The effectivity of the algorithm is presented in Section 6. We conclude the paper in Section 7.

2 Traffic Network Flows

In this section, we introduce the concept of conflict graphs that allow us to apply flooding techniques to the torus in a natural way. We start introducing the conflict graph where links of the torus are the set of vertices and two links e_1 and e_2 are adjacent if and only if they are in conflict, i.e., increasing the queue length in e_1 reduces the queue length in e_2 . Next we introduce the traffic network and green time shifts that we use to define the forward and backward flows. Finally, we introduce the conflict cycles.

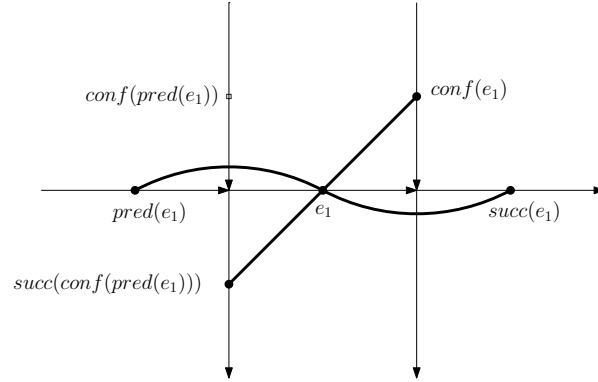


Figure 1: Conflict Graph.

Definition 2. (Conflict Graph) *The conflict graph $L(G)$ of an oriented torus $G = (V, E)$; see Figure 1, is the graph where E is the set of vertices and any two links e_1 and e_2 are adjacent if and only if one of the following conditions holds:*

1. $e_2 = \text{succ}(e_1)$ (Equivalent $e_1 = \text{pred}(e_2)$).
2. $\exists v : \{e_1, e_2\} \subseteq E^{\text{in}}(v)$; i.e., $e_2 = \text{conf}(e_1)$.
3. $\exists v : \{e_1, e_2\} \subseteq E^{\text{out}}(v)$; i.e., $e_2 = \text{bconf}(e_1)$).

Given a link $e \in E$, we denote the set of forward neighbors of e as $N^+(e) = \{\text{succ}(e), \text{conf}(e)\}$ and the set of backward neighbors as $N^-(e) = \{\text{pred}(e), \text{bconf}(e)\}$. Following, we use green time shifts instead of green time assignments to manipulate simultaneously the green time in the incoming links of a junction. We observe that the definition of conflict graphs can be easily extended to other digraphs.

Definition 3 (Traffic Network). *Let $L(G)$ be the conflict graph of $G = (V, E)$ with a deployment $W(r)$ at round r and round length Ψ . A traffic network is defined by the tuple $TN(r) = (L(G), W(r), S(r))$ where $S(r)$ is the set of green time shifts such that for all $e \in E$, $s_e(r) \in S(r)$, $-\frac{\Psi}{2} < s_e(r) < \frac{\Psi}{2}$ and integers, and for all $v \in V$, $\sum_{e \in E^{\text{in}}(v)} s_e(r) = 0$.*

Throughout the paper, we simple refer to the traffic network as $TN = (L(G), W, S)$ if it is clear from the context.

We can create a flow in the traffic network starting at any link e by applying a valid green time shift to one of its endpoints. We consider two types of flows: 1) forward flow that applies a valid shift to e and 2) backward flow that applies a valid shift to $\text{pred}(e)$. The following definition formalizes the flows.

Definition 4. (Flows in Traffic Network) *The forward flow operation $f_e^+(x; S)$ increases the shift $s_e \in S$ by x and decreases the shift $s_{\text{conf}(e)} \in S$ by x .*

Analogously, the backward flow operation $f_e^-(x; S)$ increases the shift $s_{pred(e)} \in S$ by x and decreases the shift $s_{conf(pred(e))} \in S$ by x . If S is clear from the context, we simple refer to them as $f_e^+(x)$ and $f_e^-(x)$.

We observe that $f_e^+(x) = f_{succ(e)}^-(x)$. Using the traffic time shifts S we can determine the new deployment equivalent to the store and forward model in Definition 1 as follows.

Definition 5. Given a deployment $W(r) = \{w_e(r) : \forall e \in E\}$ at round r and a shift set $S(r) = \{s_e(r) : \forall e \in E\}$, the deployment of e at round $r + 1$ is given by $w_e(r + 1) = w_e(r) - \min(\psi/2 + s_e(r), w_e(r)) + \min(\psi/2 + s_{pred(e)}(r), w_{pred(e)}(r))$.

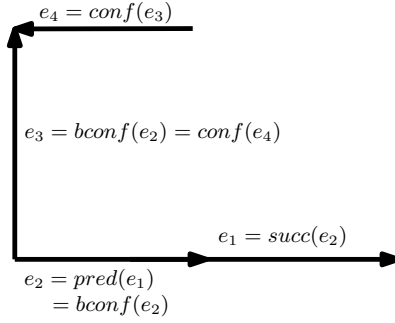


Figure 2: A conflict path (bold arrows).

Next we introduce the concept of conflict path in TN ; refer to Figure 2. We say that a path $P = \{e_1, e_2, \dots, e_l\}$ in TN is a conflicting path if $|P| > 2$ and for every three consecutive links $e_j, e_{j+1}, e_{j+2} \in P$, exactly one of the following statement follows:

1. if $e_{j+1} = conf(e_j)$, $e_{j+2} \in N^-(e_{j+1})$
2. if $e_{j+1} = succ(e_j)$, $e_{j+2} \in N^+(e_{j+1})$
3. if $e_{j+1} = pred(e_j)$, $e_{j+2} \in N^-(e_{j+1})$
4. if $e_{j+1} = bconf(e_j)$, $e_{j+2} \in N^+(e_{j+1})$

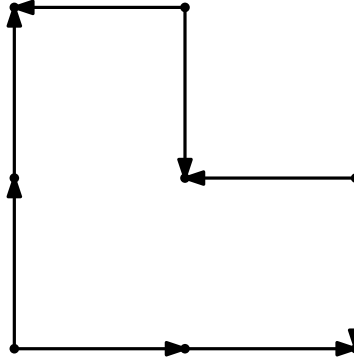


Figure 3: Conflict cycle (arrows denote the conflict cycle).

In other words, a conflict path is a collection of connected links such that there is no green time shift assignment that can reduce the agent deployment on all links. We define a conflict cycle C of length l to be

a close conflict path, i.e., $C = \{e_0, e_1, \dots, e_{l-1}, e_0\}$ such that if all links in C were undirected then C would be a cycle; see Figure 3. A 2-edge conflict subgraph of TN is a subgraph where every link is in a conflict cycle; refer to Figure 4.

We say that a link $e \in E$ is an exit link if e is in C but $\text{succ}(e)$ is not in C . Similarly, a link $e \in E$ is an entry link if e is not in C , but $\text{succ}(e)$ is in C . A vertex $v \in C$ is an exit cycle if both links in $E^{\text{out}}(v)$ are exit links. A vertex $v \in C$ is an entry cycle if both links in $E^{\text{in}}(v)$ are entry links. Let $V^-(C)$ and $V^+(C)$ denote the set of cycle exits and the set of cycle entries in C , respectively. Let \mathcal{C} denote the set of all conflict cycles in TN .

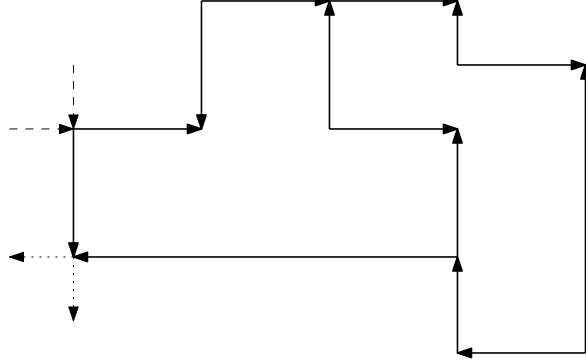


Figure 4: 2-edge conflict subgraph

Lemma 1. *In any conflict cycle C , the number of entry vertices is equal to the number of exit vertices, i.e., $|V^+(C)| = |V^-(C)|$.*

Proof. Let $C = \{e_0, e_1, \dots, e_{|C|-1}, e_0\}$. If $|V^-(C)| = 0$, then C is either a vertical or a horizontal road since for each two links e_i, e_{i+1} in C , it holds that either $e_i = \text{succ}(e_{i+1})$ or $e_i = \text{pred}(e_{i+1})$. Therefore, $|V^+(C)| = 0$.

Assume then that $|V^-(C)| > 0$. Consider any consecutive conflict path P of C in a road. Let u, v be the intersection at the end points of P . We show that u is an entry vertex and v is an exit vertex. Indeed, if that were not the case, P would have distinct directions. Thus, it contradicts the fact that each road has a unique direction. The lemma follows since each section of C in a road is delimited by one entry vertex and one exit vertex. \square

3 Lower Bound

In this section we present the lower bound on the number of agents that any algorithm can attain in each round. First we consider a saturated network and present the lower bound that any algorithm can attain. Recall that a saturated network is a TN where $w_e(r) > \psi$ for each $e \in TN$. Then, we extend the lower bound to a more general case where the network is not saturated.

Observe that a conflict cycle is, indeed, a deadlock (or gridlock in the traffic signal literature) of any arbitrary size. Thus, the number of agents in the cycle will remain constant in saturated networks in every round. Therefore, the best that any algorithm can do is to uniformly distribute the agents along the cycle. This is shown in the following theorem.

Theorem 1. Let $TN = (L(G), W, S)$ be a saturated traffic network. Then the longest queue length ϕ that any algorithm can attain is at least

$$\phi(r+1) \geq \max_{C \in \mathcal{C}} \left(\left\lceil \frac{\sum_{e \in C} w_e(r)}{|C|} \right\rceil \right).$$

Proof. Consider an optimal green time assignment A . Since TN is saturated, $w_e(r) \geq g_e(r)$ for all link $e \in TN$. Recall that $w_e(r)$ is the number of agents in e at round r and $g_e(r)$ is the green time assignment of e at round r . Let us consider any conflict cycle $C \in \mathcal{C}$. We show that the number of agents in C remains constant in the next round, i.e., $\sum_{e \in C} w_e(r) = \sum_{e \in C} w_e(r+1)$. Let $P^i = e_1^i, e_2^i, \dots, e_{l_i}^i$ be the l_i contiguous links in the same road of C . Thus, the cycle is formed by the ring segments P^i that alternate between horizontal and vertical roads. Let us consider P^i and let $W(P^i, r+1)$ be the number of agents at round $r+1$ in P^i . Then, $W(P^i, r+1) = \sum_{j=1}^{|P^i|} w_{e_j^i} - g_{e_j^i} + g_{pred(e_j^i)} = \sum_{j=1}^{|P^i|} w_{e_j^i} + g_{pred(e_j^i)} - g_{e_{|P^i|}^i}$. Thus, the number of agents in P^i changes only due to the end links.

We can calculate the number of agents in C at round r by summing the agents in each segment. In other words, $\sum_{i=1}^n W(P^i, r+1)$. To show that $\sum_{i=1}^n W(P^i, r) = \sum_{i=1}^n W(P^i, r+1) = \sum_{i=1}^n \sum_{j=1}^{|P^i|} w_{e_j^i} + g_{pred(e_j^i)} - g_{e_{|P^i|}^i}$, consider P^i and P^{i+1} . Two cases can occur:

- P^i and P^{i+1} have incoming conflict links. W.l.o.g. assume that $e_1^{i+1} = conf(e_{|P^i|}^i)$. Therefore,

$$g_{e_1^{i+1}} = \Psi - g_{e_{|P^i|}^i}.$$

- P^i and P^{i+1} have outgoing conflict links. W.l.o.g. assume that $e_{|P^i|}^{i+1} = bconf(e_1^i)$. Therefore,

$$g_{pred(e_{|P^i|}^{i+1})} = \Psi - g_{pred(e_1^i)}.$$

Thus, the number of agents in the conflict cycle is $\sum_{i=1}^n W(P^i, r+1) = \sum_{i=1}^n \sum_{l=1}^{|P^i|} w_{e_l^i} + (|V^{in}(C)| - |V^{out}(C)|)\Psi$. Further, From Lemma 1, $|V^{in}(C)| = |V^{out}(C)|$. Therefore, $\sum_{i=1}^n W(P^i, r) = \sum_{i=1}^n W(P^i, r+1)$. By the pigeon hole principle, the longest queue in P in round $r+1$ is at least $\left(\left\lceil \frac{\sum_{i=1}^n W(P^i, r)}{|C|} \right\rceil \right)$.

The theorem follows by taking the cycle with the maximum average number of agents. \square

From Theorem 1, any greedy algorithm that allows Ψ agents to cross, regardless of the direction, can only attain a locally optimal value. However, since each road has kn agents, the social optimum is k . The question then arises for which round size we can optimally solve the problem. Observe that any algorithm that attains an optimal solution must assign to some links a green time greater than the number of agents in these links. In the next lemma we establish a minimum round length necessary to compute an optimal solution using a counterexample. This value provides a threshold when the network becomes saturated.

Lemma 2. $\Psi > k - n + 1$.

Proof. We will construct a counterexample where $\Psi \geq k - n + 1$ such that every link is saturated. Let kn be the total number of agents in each road R_a . Let $e_1, e_2, e_3, \dots, e_n$ be the links in road R_a . Consider any conflict cycle C that has $n-1$ links in R_a . Let $e_j = R_a \setminus C$ and let $C_{r_a} = \bigcup_{i \neq j} e_i$ be the links of R_a in C . Consider a deployment at round r where $w_{e_j} = k - n + 1$ and $w_{e_i} = k + 1$ for all $e_i \in C_{r_a}$. Thus, the number of agents in R_a is $k - n + 1 + (n-1)(k+1) = kn$. However, since $\Psi \geq k - n + 1$, every link e in R_a has at least Ψ agents. The lemma follows from Theorem 1. \square

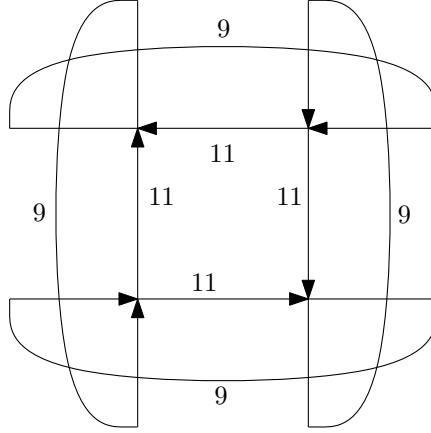


Figure 5: Traffic network with 4 roads and 20 agents in each road that requires rounds of length at least 10. Observe that the round must be at least 10 to reduce the queue of the links with 11 agents.

An example with $n = 2$ and $k = 11$ that cannot be optimally solved with round length $\psi < 10$ is depicted in Figure 5. Observe that every link has at least 9 agents. Therefore, to obtain an optimal deployment, the green time of at least one link must be greater than 9.

Next, we provide a lower bound on the queue length that any algorithm can achieve. Consider a cycle C with the maximum average number of agents and a set of entry links to C . Observe that an algorithm could introduce at most $w_e < \psi$ agents into the cycle for each entry link e . On the other hand, the number of agents that leave every cycle is at most ψ for each cycle exit. We use these observations in the following theorem to show a lower bound for unsaturated networks. The proof is similar to the proof of 1 and we leave it as an exercise to the reader.

Theorem 2. *Let $TN = (L(G), W, S)$ be a traffic network, let C be the set of conflict cycles in TN and $E^{in}(C)$ be the set of entry links of C in TN . Then,*

$$\phi(r+1) \geq \max_{C \in \mathcal{C}} \left(\left\lceil \frac{\sum_{e \in C \cup E^{in}(C)} w_e}{|C|} \right\rceil \right).$$

4 Minimizing the Longest Queue Length in Saturated Traffic Networks

In this section, we present an algorithm that minimizes the maximum queue length in saturated networks. A ϕ -conflict path P is a conflict path where for all $e \in P$ it holds that $\phi = w_e$. A ϕ -conflict tree consists of ϕ -conflict path without cycles. A ϕ -conflict cycle C is a cycle where the total number of agents in C is at least $|C|(\phi - 1)$ and at most $|C|\phi$; see Figure 6.

Next, we present an algorithm that given a link e and a capacity ϕ , verifies whether there exists a shift that induces a conflict ϕ -tree with root at e . The idea is that if the number of agents in a link e is greater than ϕ , then a flow needs to be sent to the neighboring links until it has at least ϕ agents which in turn can provoke a chain effect.

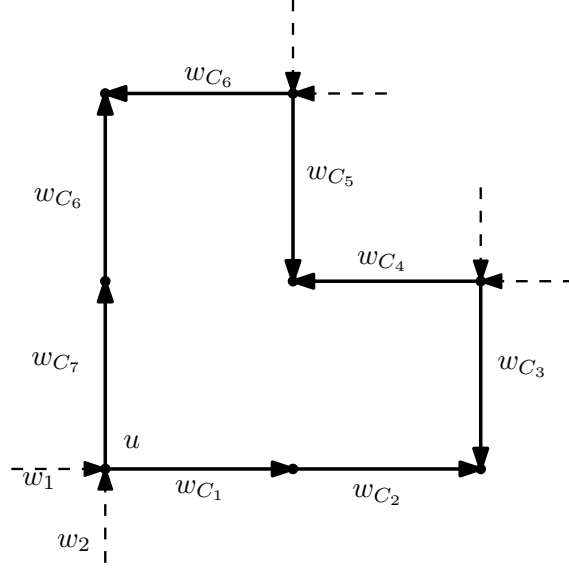


Figure 6: Characterization of the optimal solution. (The bold arrows denote the conflict cycle C and dashed arrows denote entry edges of C .)

Lemma 3. *Given a traffic network, a root link r , a direction dir and a capacity ϕ , Algorithm 1 computes in $O(n)$ expected time a new shift S , if it exists, that induces a maximal ϕ -conflict tree Π rooted at r such that $s_e \leq \psi/2$ for every link $e \in \Pi$ and if $|\Pi| > 1$, at least one of $N^{dir}(r)$ is in Π .*

Proof. The algorithm uses a stack Q to maintain the links to be processed and an array Π where $\Pi[e]$ denotes the processed link that leads to e . We define $T = (E', F)$ where $E' \subseteq E$ and $(e_2, e_1) \in F$ if and only if $e_1 = \Pi[e_2]$. We prove that T is a tree. Let l denote the diameter of T . We prove by induction on l . The inductive hypothesis is $T \setminus Q$ is acyclic, $w_e = \phi$ for all e in $T \setminus Q$, and $w_e > \phi$ for all $e \in Q$.

For the base case we assume that the diameter of T is 1. Since $\phi < w_e$, $root$ is inserted into Q . Let e be the next link extracted from Q . Observe that e is, indeed, $root$. W.l.o.g. assume that $dir = +$. The case when $dir = -$ is analogous. We decrease w_{root} by ϕ by sending a forward flow of $w_{root} - \phi$, i.e., $f_{root}^+(w_{root} - \phi)$. The forward flow affects only the neighbors in $N^+(root)$. Since $root$ is the first processed link, $succ(root)$ and $conf(root)$ have not been previously processed. However, if $w_{succ(root)} > \phi$, then $succ(root)$ violates the condition. Therefore, we insert it into Q and sets $root$ as the predecessor of $succ(root)$. Similarly, if $w_{conf(root)} > \phi$, then $conf(root)$ is inserted into Q and the predecessor of $conf(root)$ is set to $root$. Observe that $T \setminus Q$ only contains the $root$. Therefore, $T \setminus Q$ is a trivial tree where $w_{root} = \phi$, and $w_e > \phi$, $\forall e \in Q$.

For the inductive step we consider T with diameter $l > 1$. Assume that the inductive hypothesis holds in $T' = T \setminus Q$. Therefore, T' is acyclic and $w_e = \phi$ for all e in T' . Let e be the next link to be processed. Since e is extracted from Q , $w_e > \phi$. Let us assume w.l.o.g. that e is processed as consequence of a flow in $N^+(e)$. Therefore, we apply a backward flow to e of $w_e - \phi$, i.e., $f_e^-(w_e - \phi)$. Observe that although the inductive hypothesis holds in T' , the applied flow affects the queue of the backward neighbors of e ; i.e., $N^-(e)$. We consider three cases:

1. Neither $pred(e)$ nor $bconf(e)$ has been processed. If $w_{pred(e)} > \phi$, it inserts $pred(e)$ into Q and sets e as the predecessor of $pred(e)$. Since T' is acyclic, $T' \cup \{pred(e)\}$ remains acyclic. Similarly, if

Algorithm 1: Flooding

Input/Output: $TN = (L(G), W, S)$: Traffic Network
Input: $root$: The root link
Input: dir : $\{+, -\}$
Input: ϕ : Maximum capacity
Output: True if exists S' that induces a ϕ -conflict tree rooted at $root$ towards $N^{dir}(root)$ such that $\forall e \in \Pi, s_e \leq \psi/2$
Output: Π : ϕ -conflict tree rooted at $root$ such that $\forall e \in \Pi; s_e \leq \psi/2$ and if $|\Pi| > 1$, at least one of $N^{dir}(root)$ is in Π

Let Q be a stack;
Let Π be a hash table where $\Pi[e]$ is the parent of e ;
 $\Pi[root] \leftarrow root$;
if $w_{root} > \phi$ **then** $push(Q, root)$ **while** Q is not empty **do**
 $e \leftarrow pop(Q)$;
 if $\Pi[e] \in N^-(e)$ **or** $dir = +$ **then**
 $f_e^+(w_e - \phi)$;
 if $(\Pi[succ(e)] \text{ and } w_{succ(e)} > \phi)$ **or** $(\Pi[conf(e)] \text{ and } w_{conf(e)} > \phi)$ **or** $s_e > \psi/2$ **then return**
 $(false, \Pi)$
 if $w_{succ(e)} > \phi$ **then**
 $push(Q, succ(e)), \Pi[succ(e)] \leftarrow e$;
 if $w_{conf(e)} > \phi$ **then**
 $push(Q, conf(e)), \Pi[conf(e)] \leftarrow e$;
 else if $\Pi[e] \in N^+(e)$ **or** $dir = -$ **then**
 $f_e^-(w_e - \phi)$;
 if $(\Pi[pred(e)] \text{ and } w_{pred(e)} > \phi)$ **or** $(\Pi[bconf(e)] \text{ and } w_{bconf(e)} > \phi)$ **or** $s_{pred(e)} < \psi/2$ **then return**
 $(false, \Pi)$ **if** $w_{pred(e)} > \phi$ **then**
 $push(Q, pred(e)), \Pi[pred(e)] \leftarrow e$;
 if $w_{bconf(e)} > \phi$ **then**
 $push(Q, bconf(e)), \Pi[bconf(e)] \leftarrow e$;
 $dir \leftarrow null$;
return $(true, \Pi)$;

$w_{bconf(e)} > \phi$, it inserts $bconf(e)$ into Q and sets e as the predecessor of $bconf(e)$. Since T' is acyclic, $T' \cup \{bconf(e)\}$ remains acyclic. It is easy to verify that $w_{e'} = \phi$ for all $e' \in T'$.

2. Either $pred(e)$ or $bconf(e)$ has been processed, say $pred(e)$. Therefore, $w_{pred(e)} = \phi$. In this case, the algorithm returns false indicating that it is not possible to create a ϕ -conflict tree T . We show that $T = T' \cup \{pred(e)\}$ does not provide a feasible solution. Observe that T has at least one cycle since $pred(e)$ has been previously visited. Let us assume that e was processed due to a forward flow in $succ(e)$. Consider the ϕ -conflict path P in T from $succ(e)$ to $pred(e)$. Then the total number of agents in $P \cup \{e\}$ is greater than $(|P| + 1)\phi$ since $w_e > \phi$. Therefore, from the pigeon hole principle, at least one link in $P \cup \{e\}$ has $\phi + 1$ agents. Hence, there does not exist a shift S that induces a tree T rooted at $root$ that includes $N^{dir}(root)$ where $w_{e'} = \phi$ for each $e' \in T$ and $s_{e'} \leq \psi/2$ for every $e' \in T \setminus Q \cup \{e\}$.
3. Either $pred(e)$ or $bconf(e)$ has been processed, say $bconf(e)$, but $w_{bconf(e)} \leq \phi$. Therefore, $bconf(e)$ is not inserted into Q and $T = T' \cup \{e\}$ remains acyclic. Similarly, if $pred(e)$ has been previously

processed and $w_{pred(e)} \leq \phi$, $pred(e)$ is not inserted into Q and $T = T' \cup \{e\}$ remains acyclic.

When Q becomes empty, T remains acyclic and for all $e \in \Pi$, $w_e = \phi$. Since links are visited once and the operations in Π and P take $O(1)$ expected time, the complexity of the algorithm is linear in the number of edges. However, since TN is planar, the number of links is linear in the number of intersections. Therefore, the running time of the algorithm is $O(n)$ expected time. \square

Observe that when Lemma 3 fails, a ϕ -conflict cycle arises. The following algorithm, which is the main result of this section, determines the minimum longest queue length in saturated traffic networks. The idea is to reduce the queue length of the longest queue length using Algorithm 1 one by one until it fails.

Theorem 3. *Let $TN = (L(G), W, S)$ be a traffic network, Algorithm 2 computes a valid assignment shift S that induces a ϕ -conflict 2-edge connected subgraph where ϕ is the length of the maximum queue length. Further, it can be computed in $O(n^2)$ expected time.*

Algorithm 2: ϕ -Conflict 2-Edge Connected Subgraph

```

Input/Output:  $TN = (L(G), W, S)$ : Traffic Network
Output:  $C$ :  $\phi$ -conflict 2-edge subgraph
Output:  $\phi$ : longest queue length
let  $\phi$  be the length of the longest queue in  $TN$ ;
let  $C = \{\emptyset\}$  be the set of  $\phi$ -conflict two-edge subgraph of  $TN$ ;
let  $PQ$  be a queue of links in  $E \setminus \mathcal{P}$  such that  $w_e = \phi$  for each  $e \in E \setminus \mathcal{P}$ ;
while  $PQ \neq \emptyset$  do
     $e \leftarrow gets(PQ)$ ;
     $S' \leftarrow S$ ;
     $\phi \leftarrow w_e$ ;
     $(state, \Pi_f) \leftarrow Flooding(TN, e, +, \phi - 1)$ ;
    if  $\neg state$  then
         $S \leftarrow S'$ ;
         $(state, \Pi_b) \leftarrow Flooding(TN, e, -, \phi - 1)$ ;
        if  $\neg state$  then
            /* A  $\phi$ -conflict cycle has been created */
             $S \leftarrow S'$ ;
            determine the  $\phi$ -conflict cycle  $C$  in  $\Pi_f \cup \Pi_b \cup C$  and add it to  $C$ ;
        update  $PQ$ ;
return  $\phi$ ;

```

Proof. The key idea of the algorithm is to decrease the length of the longest queue in the traffic network by sending a flow to the neighbors one by one until it cannot be reduced anymore. Let C be the set of ϕ -conflict two-edge connected subgraph and ϕ be the length of the longest queue in TN . Let PQ be a queue of links in $E \setminus C$ such that for all $e \in PQ$, $w_e = \phi$.

The invariant that we maintain is that for each link $e \in PQ$ such that $w_e = \phi$, e is either in PQ or in a ϕ -conflict cycle in C . Observe that when PQ is empty, then the shift S induces a set C of ϕ -conflict cycles.

Consider $e \in PQ$ such that w_e is maximum. In case of tie, we break it arbitrarily. Let $\phi = w_e$. To reduce the queue length of e , we use the flooding algorithm depicted in Algorithm 1 to send a flow unit

to either its forward neighbors or its backward neighbors. Observe that when Algorithm 1 returns *true*, S induces a ϕ -conflict tree rooted at e . However, when it returns *false*, S is infeasible. From the minimality of Algorithm 1, ϕ cannot be reduced since a cycle is created.

The algorithm keeps the last valid shift in S' and tries to reduce the queue length one by one in e by sending a flow unit to its forward neighbors. If it succeeds, the queue length of each link in Π_f has $\phi - 1$ agents. However, if it fails, it tries to reduce the queue length in e by sending a flow unit to its backward neighbors with the last valid shift S' . If it succeeds, the queue length of each link in Π_b has $\phi - 1$ agents. However, if the backward flow also fails, e is in a ϕ -conflict cycle P . Therefore, the minimum maximum queue length is ϕ . We add to PQ every link $e' \in PQ$ in $E \setminus \mathcal{P}$ such that $w_{e'} \geq \phi$. Therefore, the algorithm finishes when every link in PQ is processed and no new links are added to P when updating. Clearly, the previous procedure maintains the invariant.

Regarding the complexity, the flooding algorithm takes $O(n)$ expected time and determining the longest queue length can be trivially implemented in linear time. Since the while loop is executed no more than $O(kn)$ iterations and, updating PQ also takes $O(n)$ time, the complete algorithm takes $O(kn^2)$ time since the ϕ -conflict cycles can be determined in $O(n)$ using BFS as the number of links is linear in the number of intersections. However, since k is a constant, the complexity of the algorithm is $O(n^2)$ time. \square

Corollary 1. *There exists an algorithm that minimizes the longest queue length in a saturated torus that runs in $O(n^2)$ time.*

Proof. From Theorem 3, we can compute ϕ -conflict cycles in a traffic network in $O(n^2)$ time. The optimality comes directly from Theorem 1. \square

5 Optimal Queue Length in Unsaturated Traffic Networks

In this section we study the problem of minimizing the longest queue length in unsaturated networks. The first question that arises is for which values of the round ψ and k the problem can be solved optimally. From Lemma 2, the problem cannot be solved if the round size is at least $k - n + 1$. Therefore, we assume throughout this section that $\psi > k - n + 1$.

First we define the rotation which allows us to increase or decrease the shift assignments of the 2-edge conflict subgraph C without increasing the queue length of the links in C . For simplicity of presentation, we assume that $s_e = 0$ for each link $e \in TN$.

Definition 6. (Rotating) *Let $C = P^0 P^1 P^2 \dots P^{2l+1}$ be the contiguous road sections of a ϕ -conflict cycle. We say that the cycle rotates for a constant a if $f_e^+(a)$ for each $e \in P^{2i}$ and $f_e^+(-a)$ for each $e \in P^{2i+1}$ for all $i \in [0, l]$.*

In the following lemma we show that a complete ϕ -conflict 2-edge subgraph C of TN can be rotated to obtain a new ϕ -conflict 2-edge subgraph without increasing the queue length of any link in C .

Lemma 4. *Given a constant a and a ϕ -conflict 2-edge subgraph C of TN such that for each $e \in C$, $|s_e| \leq \psi/2 - a$, we obtain a new ϕ' -conflict 2-edge subgraph of TN by rotating at most a ; where $\phi' \leq \phi$.*

Proof. Observe that cycles consist of alternating horizontal and vertical roads; see Figure 7. Thus, we can apply either a positive flow to horizontal roads and negative flow to vertical roads or a negative flow to horizontal roads and positive flow to vertical roads. From Lemma 1, the queue length of each link in the ϕ -conflict 2-edge subgraph does not increase. Consider an entry link e such that $w_e < g_e + a$. W.l.o.g. assume that $w_{succ(e)} = \phi$ where $succ(e)$ is in the ϕ -conflict 2-edge subgraph. Therefore, after a rotation of a ,

$w_{succ(e)}(r+1) = w_{succ(e)}(r) - (g_e + a) + w_e(r) = \phi + w_e(r) - (g_e + a) < \phi$ since $w_e < g_e + a$. We observe that after a rotation, the queue length of some links not in the ϕ -conflict 2-edge subgraph can increase greater than ϕ . However, we can easily verify whether the length can be reduced using flooding Algorithm 1. The lemma easily follows since roads have consistent directions. \square

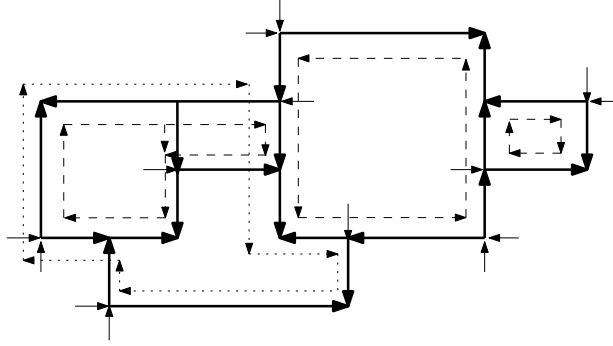


Figure 7: Rotation of a ϕ -conflict 2-edge subgraph of TN .

We are ready to provide the main result of this section. The algorithm essentially reduces the queue length of all entry links to the minimal queue length by sending consecutive backward flows, and rotates the ϕ -conflict 2-edge subgraph in such a way that the average number of agents in the ϕ -conflict 2-edge subgraph reduces.

Theorem 4. *There exists an algorithm that given an unsaturated traffic network TN determines the minimum longest queue length.*

Proof. Consider Algorithm 3. Let C be the ϕ -conflict two-edge subgraph of $L(G)$ using Algorithm 1. To prove that eventually the strategy leads to a social optimum, we show that either the number of ϕ -conflict cycles in C increases or the number of agents in C decreases. Let e' be any link in C with queue length ϕ and let e be the first entry link that is found in the same road.

We reduce the queue length of e such that $w_e < \phi$. We claim that such a link exists. Indeed, if the queue length is not optimal, at most $n - 1$ links have $k + 1$ agents while one link has $k - (n - 1)$ agents. However, by the assumption $k - n + 1 < \psi$. Two cases can occur:

- $w_e \geq \psi$, i.e. we cannot reduce the queue length of e . Therefore, we can extend C since a ϕ -conflict cycle prevents e to be reduced.
- $w_e < \psi$. Assume that there exists a rotation of C such that $s_e \leq \psi/2$ and the length queue of each link is at least ϕ . In other words, while rotating C , some links have queue length greater than ϕ . But, the queue length can be reduced using Algorithm 1. Therefore, the total number of agents in C reduces by 1. However, while rotating, a link not in C cannot be reduced using Algorithm 1, we can extend C as a new ϕ -cycle is found.

This completes the proof of the theorem as eventually the queue length will be reduced. \square

Algorithm 3: Minimizing the Longest Queue Length

Input/Output: $TN = (L(G), W, S)$: Traffic Network
 $C \leftarrow \emptyset$;
while $C \neq L(G)$ **do**
 let C be the ϕ -conflict two-edge subgraph of $L(G)$ using Algorithm 1;
 $reduce \leftarrow False$;
 while $\neg reduce$ **do**
 let e be the entry link to C adjacent to a cycle in C such that the road section incident to e has at least a link of queue length ϕ ;
 $cycleFound \leftarrow False$;
 if $release(TN, e, \phi)$ **then**
 $a \leftarrow 0$;
 while $\psi/2 + a \leq w_e + 1$ **do**
 rotate C so that $s_e = a$;
 let E' be the set of links in $E \setminus C$ such that $\forall e' w_{e'} > \phi$;
 $cycleFound \leftarrow False$;
 foreach $e' \in E'$ **do**
 if $succ(e') \in C$ **or** $conf(e) \in C$ **then** $dir \leftarrow -$ **else** $dir \leftarrow +$
 $(noCycle, \Pi) \leftarrow Flooding(TN, e', dir, \phi)$;
 if $\neg noCycle$;
 then
 determines the ϕ -conflict cycles in $\Pi \cup e' \cup C$ and add them to C ;
 $cycleFound = True$;
 if $cycleFound$ **then**
 break;
 if $\neg cycleFound$ **then**
 $reduce = True$;
Function $release(e)$
 Backup S ;
 while $w_e > \psi - 1$ **do**
 $f_e^-(1)$;
 if $w_{pred(e)} > \phi$ **then**
 $(noCycleb, \Pi_b) \leftarrow Flooding(TN, pred(e), -, \phi)$;
 $(noCyclef, \Pi_f) \leftarrow Flooding(TN, bconf(e), +, \phi)$;
 if not $noCycleb$ **or not** $noCyclef$ **then**
 Restore S ;
 determines the ϕ -conflict cycles in $\Pi_b \cup \Pi_f \cup C \cup \{e\}$ and add them to C ;
 return $False$;
return $True$;

6 Experiments

We implemented the Algorithm 2 in python and show that the algorithm reduces the longest queue length in torus networks where the rings have any arbitrary number of cars. We tested the program in a torus network of dimension 5×5 . We assigned to each link a random number of cars uniformly distributed in $[10, 20]$. We

set the phase size to $\psi = 26$. The minimum and maximum number of cars in the initial deployment in the queues was 7 and 21, respectively. The left torus in Figure 8 depicts the initial deployment. We ran the program for one phase and a conflict cycle was found. The longest queue was reduced to 17 and the shortest queue length increased to 7. The right torus in Figure 8 depicts the network after one phase.

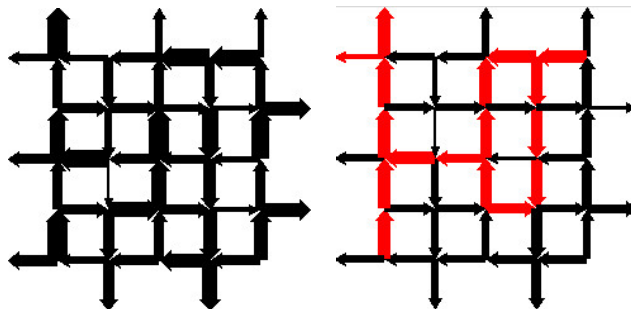


Figure 8: Torus of dimension 5x5. The length of the queues is represented with the thickness of the link. The conflict graph is drawn in red.

7 Conclusion

In this paper we have studied the problem of determining the minimum longest queue length in a torus in saturated and unsaturated networks. We focus on the fundamental questions of lower bounds and upper bounds on queue lengths when the agents move in a straight line. We did not address the number of rounds that are needed to solve the problem. Thus, the immediate open problem is to determine the minimum number of rounds required to solve Problem 1 and 2. Our paper opens a new research direction for the problem of mitigating traffic congestion in urban areas. One direction is to study the effects when the model becomes probabilistic and, for example includes turning rates and agents that appear and disappear. Another direction is to provide distributed protocols that minimizes the longest queue lengths.

References

- [1] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. Traffic light control in non-stationary environments based on multi agent q-learning. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1580–1585. IEEE, 2011.
- [2] K Aboudolas, M Papageorgiou, and E Kosmatopoulos. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 17(2):163–174, 2009.
- [3] K Aboudolas, M Papageorgiou, A Kouvelas, and E Kosmatopoulos. A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 18(5):680–694, 2010.

- [4] Konstantinos Aboudolas, Markos Papageorgiou, and Elias Kosmatopoulos. Control and optimization methods for traffic signal control in large-scale congested urban road networks. In *American Control Conference, 2007. ACC'07*, pages 3132–3138. IEEE, 2007.
- [5] I Arel, C Liu, T Urbanik, and AG Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *Intelligent Transport Systems, IET*, 4(2):128–135, 2010.
- [6] R Barlović, Thorsten Huisinga, Andreas Schadschneider, and Michael Schreckenberg. Adaptive traffic light control in the chsch model for city traffic. In *Traffic and Granular Flow'03*, pages 331–336. Springer, 2005.
- [7] Alberto Bemporad, Francesco Borrelli, and Manfred Morari. Min-max control of constrained uncertain discrete-time linear systems. *Automatic Control, IEEE Transactions on*, 48(9):1600–1606, 2003.
- [8] Peter J Campo and Manfred Morari. Robust model predictive control. In *American Control Conference, 1987*, pages 1021–1026. IEEE, 1987.
- [9] Eduardo Camponogara and Helton F Scherer. Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints. *Automation Science and Engineering, IEEE Transactions on*, 8(1):233–242, 2011.
- [10] Donald Watts Davies. The control of congestion in packet-switching networks. *Communications, IEEE Transactions on*, 20(3):546–550, 1972.
- [11] Lucas Barcelos de Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.
- [12] Carlos Gershenson. Self-organizing traffic lights. *arXiv preprint nlin/0411066*, 2004.
- [13] Paweł Gora. A genetic algorithm approach to optimization of vehicular traffic in cities by means of configuring traffic lights. In *Emerging Intelligent Technologies in Industry*, pages 1–10. Springer, 2011.
- [14] Peter B Key and Derek R McAuley. Differential qos and pricing in networks: Where flow control meets game theory. *IEE Proceedings-Software*, 146(1):39–43, 1999.
- [15] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996.
- [16] S. Lämmer and D. Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(04):P04019, 2008.
- [17] Long Bao Le, Eytan Modiano, Changhee Joo, and Ness B Shroff. Longest-queue-first scheduling under sinr interference model. In *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, pages 41–50. ACM, 2010.
- [18] Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Fast model predictive control for urban road networks via milp. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):846–856, 2011.

- [19] Gregory Stephanopoulos, Panos G Michalopoulos, and George Stephanopoulos. Modelling and analysis of traffic queue dynamics at signalized intersections. *Transportation Research Part A: General*, 13(5):295–307, 1979.
- [20] Tamás Tettamanti, Tamas Luspay, Balázs Kulcsár, Tamás Péni, and István Varga. Robust control for urban road traffic networks. *Intelligent Transportation Systems, IEEE Transactions on*, 15(1):385–398, 2014.
- [21] Tamás Tettamanti, István Varga, Balázs Kulcsár, and József Bokor. Model predictive control in urban traffic network management. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1538–1543. IEEE, 2008.
- [22] Sam Toueg and Kenneth Steiglitz. Some complexity results in the design of deadlock-free packet switching networks. *SIAM Journal on Computing*, 10(4):702–712, 1981.
- [23] Bao-Lin Ye, Weimin Wu, and Weijie Mao. Distributed model predictive control method for optimal coordination of signal splits in urban traffic networks. *Asian Journal of Control*, 2014.